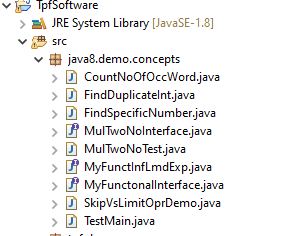
Java 8 Programming

1.



1. Program

**package** java8.demo.concepts;

**import** java.util.Arrays;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.function.Function;

**import** java.util.stream.Collectors;

/\*\*

\* Class CountNoOfOccWord

\*

\* **@author** Nikhil Kapaley

\*

\* Q: Count the number of occurrence of words in given string using

\* java8 ?

\*

\* Input: "welcome to Java eight programming world welcome you all."

\*

\* Output:

\* {welcome=2,to=1,Java=1,eight=1,programming=1,world=1,you=1,all=1}

\*/

**public** **class** CountNoOfOccWord {

**public** **static** **void** main(String[] args) {

String str = "welcome to Java eight programming world welcome you all";

List<String> strList = Arrays.*asList*(str.split(" "));

Map<String, Long> map = strList.stream()

.collect(Collectors.*groupingBy*(Function.*identity*(), Collectors.*counting*()));

System.***out***.println(map);

// Example: Function will using static method called identity() and this

// identity will return what we are passing in the identity

/\*

\* Function<String, String> fn = Function.identity();

\*

\* System.out.println(fn.apply("Hello Nikhil"));

\*/

}

}

Output:

{all=1, Java=1, world=1, to=1, welcome=2, you=1, programming=1, eight=1}

2 program

**package** java8.demo.concepts;

**import** java.util.Arrays;

**import** java.util.HashSet;

**import** java.util.List;

**import** java.util.Set;

**import** java.util.stream.Collectors;

/\*\*

\* Class FindDuplicateInt

\*

\* **@author** Nikhil Kapaley

\*

\* Q: Find the duplicate elements in a given integers list using java8

\* using stream function?

\*

\* Input:[10,20,15,15,12,8,19,23,20,25,10]

\*

\* Output:10,20,15

\*/

**public** **class** FindDuplicateInt {

**public** **static** **void** main(String[] args) {

// create a list of integer

List<Integer> intList = Arrays.*asList*(10, 20, 15, 15, 12, 8, 19, 23, 20, 25, 10);

// create a set of integer

Set<Integer> intSet = **new** HashSet<>();

// apply stream operation over a intList using filter list of elements predicate

// and consumer interface apply here

intList.stream().filter(x -> !intSet.add(x)).collect(Collectors.*toSet*())

.forEach(x -> System.***out***.println("duplicate element is:" + x));

}

}

Output:

duplicate element is:20

duplicate element is:10

duplicate element is:15

1. Program

**package** java8.demo.concepts;

**import** java.util.Arrays;

**import** java.util.List;

/\*\*

\* Class FindSpecificNumber

\*

\* **@author** Nikhil Kapaley

\*

\* Q: Write a program for find the number which starts with 1 only and

\* give output?

\*

\*/

**public** **class** FindSpecificNumber {

**public** **static** **void** main(String[] args) {

List<Integer> lists = Arrays.*asList*(1, 2, 12, 3, 5, 15, 100, 50, 168, 169, 170);

lists.stream().filter(t -> t.toString().startsWith("1")).forEach(System.***out***::println);

}

}

Output:

1

12

15

100

168

169

170

1. Program

**package** java8.demo.concepts;

**import** java.util.Arrays;

**import** java.util.List;

/\*\*

\* Class SkipVsLimitOprDemo

\*

\* **@author** Nikhil Kapaley

\*

\* Q: What is the difference between limit() and skip() using example?

\*

\* The skip(n) method is an intermediate operation that discards the

\* first n elements of a stream. The n parameter can't be negative, and

\* if it's higher than the size of the stream, skip() returns an empty

\* stream.

\*

\* The limit(n) method is another intermediate operation that returns a

\* stream not longer than the requested size. As before, the n parameter

\* can't be negative.

\*

\*/

**public** **class** SkipVsLimitOprDemo {

**public** **static** **void** main(String[] args) {

// create a list of integer

List<Integer> lists = Arrays.*asList*(10, 20, 30, 35, 40, 45, 50, 55, 60, 65, 70);

// limit operation I want to limit this list to 5 only

lists.stream().limit(5).forEach(System.***out***::println);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*limit vs skip\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

// skip operation I have to skip this list till 6

lists.stream().skip(6).forEach(System.***out***::println);

}

}

Output:

10

20

30

35

40

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*limit vs skip\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

50

55

60

65

70